

CTR

Extended Specifications for the DMPGL 2.0 POD System API

2012/02/10

Version 1.3

**The content of this document is highly confidential
and should be handled accordingly.**

Confidential

These coded instructions, statements, and computer programs contain proprietary information of Nintendo and/or its licensed developers and are protected by national and international copyright laws. They may not be disclosed to third parties or copied or duplicated in any form, in whole or in part, without the prior written consent of Nintendo.

Table of Contents

1	Overview	5
1.1	API	5
1.1.1	POD Initialization	5
1.1.2	POD Shutdown.....	5
1.1.3	Checking the Environment for Using POD	5
1.1.4	Setting a Dummy POD Allocator	6

Code

Code 1-1	dmpglInitializePicaOnDesktop	5
Code 1-2	dmpglFinalizePicaOnDesktop.....	5
Code 1-3	dmpglCheckPicaOnDesktopAvailable	5
Code 1-4	dmpglSetDummyAllocator	6

Revision History

Version	Revision Date	Description
1.3	2012/02/10	Updated copyright date.
1.1	2010/08/06	- Added section 1.1.3 Checking the Environment for Using POD. - Added section 1.1.4 Setting a Dummy POD Allocator.
1.0	2009/11/30	Initial version.

1 Overview

This document explains the DMPGL 2.0 POD system API.

PicaOnDesktop (POD) uses the WGL library to control the display. To run an application with POD, you must initialize and shut down POD using functions that are separate from the WGL API.

1.1 API

This section describes each function in the API.

1.1.1 POD Initialization

Code 1-1 dmpglInitializePicaOnDesktop

```
GLboolean dmpglInitializePicaOnDesktop (void);
```

Initializes POD. You must initialize POD with this function before using any other function from the DMPGL 2.0 API. Before calling this function, though, you must configure the current context of its calling thread to be a valid WGL context.

This function returns `GL_TRUE` when initialization succeeds and `GL_FALSE` when initialization fails. When a valid WGL context has not been set, this function may internally be forced to exit.

This function may succeed in initialization even on a computer with a graphics card that does not meet the conditions for using POD. If so, behavior will be undefined when you run a feature that is lacking. Use the `dmpglCheckPicaOnDesktopAvailable` function to determine whether the conditions for using POD have been met.

1.1.2 POD Shutdown

Code 1-2 dmpglFinalizePicaOnDesktop

```
Void dmpglFinalizePicaOnDesktop (void);
```

Shuts down POD. Behavior is not guaranteed for any DMPGL 2.0 functions that are called after this one. To use DMPGL 2.0 functions, call `dmpglInitializePicaOnDesktop` again after this function.

1.1.3 Checking the Environment for Using POD

Code 1-3 dmpglCheckPicaOnDesktopAvailable

```
GLboolean dmpglCheckPicaOnDesktopAvailable (const char** report);
```

Checks the environment in which POD is used. This checks the graphics card features on the computer running POD and determines whether the features used by POD are supported. A value of `GL_TRUE` is returned if the conditions for using POD have been met and `GL_FALSE` is returned if some features are unsupported. When `GL_FALSE` is returned, a pointer to a string with the names of the missing functions and extended features is returned in `report`.

Call this function from a thread that has a valid WGL context set as its current context. You cannot get correct information when a valid WGL context has not been set.

You can use this function both before and after you call the `dmpglInitializePicaOnDesktop` function.

1.1.4 Setting a Dummy POD Allocator

Code 1-4 `dmpglSetDummyAllocator`

```
void dmpglSetDummyAllocator (
    void* (*allocator)(GLenum, GLenum, GLuint, GLsizei),
    void (*deallocater)(GLenum, GLenum, GLuint, void*));
```

Sets a dummy POD allocator. The allocator and deallocater set by this function are not meant to allocate memory that is actually used by POD; they simply simulate calls to the allocator and deallocater under the same timing and conditions as the actual hardware. POD does not use the return value from `allocator`. For more details on the arguments passed to `allocator` and `deallocater` and on the return value from `allocator`, see the section on DMPGL initialization in the *DMPGL 2.0 System API Specifications*.

By default, `allocator` and `deallocater` are set equal to `NULL`. The allocator and deallocater are not called when these arguments are `NULL`.

The allocator and deallocater are only called for texture memory, vertex buffer memory, and render buffer memory. They are not called for system memory, display buffer memory, and 3D command buffer memory.

Functions such as `glTexImage2D`, `glBufferData`, and `glRenderbufferStorage` can specify how to transfer data as well as where to place it in memory for POD, just as they can for the actual hardware. The specified settings only affect calls to this function's allocator and deallocater; they do not affect how data is manipulated for POD itself. For more details on how to manipulate data, see the *DMPGL 2.0 Data Manipulation Specifications*.

DMP and PICA are registered trademarks of Digital Media Professionals Inc.

All company and product names in this document are the trademarks or registered trademarks of their respective companies.

© 2009–2013 Digital Media Professionals Inc. All rights reserved.

This documentation is the confidential and proprietary property of Digital Media Professionals Inc. And the possession or use of this documentation file and contained information requires a written license from Digital Media Professionals Inc.

© 2009–2013 Nintendo

The contents of this document cannot be duplicated, copied, reprinted, transferred, distributed, or loaned in whole or in part without the prior approval of Nintendo.